Objetos y JSON

Teoría

Cómo crear un objeto en javascript?

Simplemente escribiendo dos llaves `{ }` ya tenemos un objeto en javascript. Es usual (pero no obligatorio) guardar los objetos dentro de alguna variable:

Ejemplo

```
const persona = {}
```

Cómo agregarle propiedades a un objeto?

Simplemente haciendo `nombreDelObjeto.nombreDeLaPropiedad = algunValor` ...:

- Si la propiedad no existía, creamos la propiedad con el valor dado.
- Si la propiedad ya existía, actualizamos su valor.

Ejemplo

```
const persona = {}
persona.nombre = 'mariano'
persona.edad = 32
```

También podemos utilizar la siguiente sintaxis:

```
const persona = {}
persona['nombre'] = 'mariano'
persona['edad'] = 32
```

Esta sintaxis nos da la ventaja de que como el argumento que pasamos entre corchetes es un string, este puede ser obtenido de diversas maneras, o incluso recibido por parámetro.

Cómo acceder a las propiedades a un objeto?

Al igual que para agregar, existen dos maneras de acceder a una propiedad de un objeto:

- Utilizando la notacion: `objeto.propiedad`
- Utilizando la propiedad: `objeto[propiedad]`

Cualquiera de las dos variantes me devuelve el mismo resultado, que es el valor de la propiedad en cuestión.

Ejemplo

```
console.log(persona.nombre)
console.log(persona['edad'])
```

Cómo quitarle propiedades a un objeto?

Para quitarle una propiedad a un objeto usaremos la palabra reservada **`delete`** y a continuación la propiedad que queremos borrar, accediéndola desde su objeto contenedor:

Ejemplo

```
delete persona.edad
```

Propiedades anidadas

Es posible declarar propiedades que sean a su vez también objetos.

Ejemplo

```
const persona = {}
persona.nombre = 'mariano'
persona.edad = 32
persona.dirección = {}
persona.direccion.calle = 'rivadavia'
persona.direccion.numero = 1234
persona.telefonos = []
persona.telefonos.push('15-1234-5678')
persona.telefonos.push('15-1234-5000')
```

Declaración e inicialización en un solo paso

Nótese que al representar al objeto (mostrarlo) éste aparece como una serie de pares de datos, separados por comas, todo dentro de las llaves que delimitan al objeto. Estos pares de datos representan siempre una clave y un valor, es decir, el nombre del par, y el valor asociado a ese nombre. El valor asociado puede ser cualquier cosa, incluyendo (no exclusivamente) numeros, strings, objetos, arrays, etc.
Es posible declarar un objeto en js ya inicializandolo con sus valores, todo en una sola operación. En este caso, se debe separar cada clave de cada valor usando dos puntos (`:`), y cada par clave/valor del siguiente usando comas (`,`).

Ejemplo

```
const persona = {
      nombre: 'mariano',
      edad: 32,
      direccion: {
          calle: 'rivadavia',
          numero: 1234
      },
      telefonos: ['15-1234-5678', '15-1234-5000']
}
```

Cómo mostrar el contenido de un objeto?

Imaginemos que tenemos un objeto:

```
const miObj = {
    unArray: [6,7,8],
    unObjeto: { }
}
```

Si queremos mostrar por pantalla un objeto de js, lo más intuitivo sería simplemente hacer:

```
console.log(miObj)
```

Si ejecutamos nuestro programa desde la terminal, la salida será:

```
{ unArray: [ 6, 7, 8 ], unObjeto: {} }
```

Sin embargo, si debuggeamos el programa, console.log solo nos mostrará el primer nivel de un objeto. Si intentaramos mostrar un objeto con otros objetos anidados, éstos no se mostrarían correctamente:

```
Object {unArray: Array(3), unObjeto: Object}
```

Si queremos mostrar todo el objeto, hasta una determinada profundidad del mismo, usaremos la función "`util.inspect( ... )`" del modulo "`util`" que debemos requerir antes de usarla.

```
const util = require('util')
```

```
const miObj = { unArray: [6,7,8], unObjeto: {} }
console.log(util.inspect(miObj, false, 2));
```

(el segundo argumento, booleano, indica si quiero mostrar propiedades ocultas del objeto, el tercer argumento, entero, indica cuántos niveles del objeto quiero mostrar).

JSON: Java Script Object Notation

JSON es una forma de representar objetos de javascript como texto. Es considerada hoy una de las formas de serialización de datos más utilizada. Es fácilmente interpretable ya que tiene la misma estructura de pares clave/valor separados por dos puntos, y esos pares a su vez separados por comas. La principal diferencia es que mientras que en los objetos de js **las propiedades** se ven como variables (sin comillas), en json **se escriben como strings**, usando comillas, y siempre comillas dobles (**"**).

Ejemplo

```
// Objeto en js:
const persona = {
      nombre: 'mariano',
      edad: 32
}
```

JSON válido

```
// forma indentada:
{
     "nombre": "mariano",
     "edad": 32
}

// forma compacta:
{ "nombre": "mariano", "edad": 32 }
```

JSON Inválido

```
// faltan comillas en las claves:
{
     nombre: "mariano",
     edad: 32
}
```

```
// las comillas de los strings deben ser dobles:
{
     'nombre': 'mariano',
     'edad': 32
}

// faltan las comas:
{
     'nombre': 'mariano'
     'edad': 32
}

// sobra la coma del último par clave/valor!:
{
     'nombre': 'mariano',
     'edad': 32,
}
```

Cómo convertir un objeto js a JSON y viceversa?

Javascript cuenta con una librería nativa JSON que nos permite tanto pasar de objeto a JSON y de JSON a objeto.

Convertir de objeto a JSON

Se utiliza la función JSON.stringify(…) que recibe un objeto y devuelve un string con la representación de ese objeto en formato JSON.

Ejemplo

```
const persona = {
      nombre: 'mariano',
      edad: 32
}
console.log(JSON.stringify(persona))
```

Salida

```
{ "nombre": "mariano", "edad": 32 }
```

Si deseamos que el string salga con la indentación característica de JSON, podemos agregar algunos parámetros a la llamada al stringify.

```
console.log(JSON.stringify(persona, null, 4))
```

Salida

```
{
      "nombre": "mariano",
      "edad": 32
}
```

(En este caso, el 4 representa la cantidad de espacios que se dejará como indentación entre nivel y nivel).

Convertir de JSON a objeto

Se utiliza la función JSON.parse(…) que recibe un string con la representación de un objeto en formato JSON y devuelve el objeto correspondiente de js.

Ejemplo

```
const persona = JSON.parse('{ "nombre": "mariano", "edad": 32 }')
console.log(persona.edad)
```

Salida

```
32
```